

Reto 3: Distribución de vacunas - Corporación Umbrella

| | |
|------------|--|
| Propósito | Verificar y validar datos con lo aprendido hasta la semana 3 en misión tic ciclo 1, (condicionales y operadores) |
| Dificultad | ★★★ |
| Fecha | @May 9, 2022 |
| Fuente | MisionTic2022 Ciclo 1 |
| # Numero | 19 |
| Adjuntos | Reto_Semana_3.pdf |

Autor: **David Santiago Urbano Rivadeneira**

Proceso Ideal

1. Identificar el problema


- ¿Cuál es el problema? → Ante la creciente demanda de las vacunas contra el Covid-19 para poder alcanzar la inmunidad de rebaño entre la comunidad se hace necesario la contratación de personal encargado de manejar los camiones que llevarán las vacunas a diferentes regiones del país. Es por ello que se debe realizar un aplicativo para facilitar el registro de los aspirantes a la base de datos de la Corporación


- Stakeholders

| Usuarios | Cliente |
|---|----------------------|
| Aspirantes al cargo para la Distribución de vacunas | Corporación Umbrella |

- Restricciones

Las variables con las que no se efectuarán operaciones se definirán como cadenas para evitar problemas de desbordamiento.

| Variable | Restricciones  |
|----------------|--|
| nombre | debe ser de tipo cadena |
| identificacion | debe estar en el rango: [10'000.000 – 100'000.000] |
| correo | debe tener el carácter '@' no más de una vez. |
| sobrenombre | El sobrenombre no debe iniciar con un valor numérico y no debe contener más de 3 veces la letra 'a'. |

| Variable | Restricciones  |
|--------------|---|
| clave | La clave de acceso debe tener al menos uno de los siguientes símbolos ['_', '&', '?'] |
| tiempo | en meses → entero |
| tratamiento | [Si, No] Bool |
| conocimiento | [Si, No] Bool |

2. Definir el problema

- ¿Que conozco? Para la contratacion es necesario las variables previamente mencionadas en las restricciones
- Se dividió el problema en subproblemas, se propone la siguiente estructura para las funciones:
 - ☐ validar_nombre(nombre)
 - ☐ validar_ident(identificacion)
 - ☐ validar_correo(correo)
 - ☐ validar_sobrenombre(sobrenombre)
 - ☐ validar_clave(clave)
 - ☐ asignar_zona(tiempo, tratamiento, conocimiento)

3. Estrategia

Ejemplos:

Entrada al programa:

```
luzma
10000000
@
luzma
-
0
Si
Si
```

Salida del programa

```
Nombre: luzma
Identificación: 10000000
Registro Exitoso! Bienvenido a la Corporación Umbrella.
Tu zona asignada para la distribución de la vacuna es: Central
Que tenga un Feliz Día!
```

4. Algoritmos

- Requisitos para cada subproblema

Requerimientos funcionales

| | |
|---------|--|
| Name | RF1: Validar Nombre |
| Summary | El aplicativo valida si el nombre es string o no |

| | |
|--------|--------|
| Inputs | |
| | nombre |

| | |
|---------|--|
| Name | RF2: Validar identificacion |
| Summary | El aplicativo valida si la identificacion cumple entre estar entre el rango 10'000.000 y 100'000.000 |
| Inputs | |
| | identificacion |

| | |
|---------|--|
| Name | RF3: Validar correo |
| Summary | Valida si el correo contiene el caracter '@' y que no se repita mas de una vez |
| Inputs | |
| | correo |

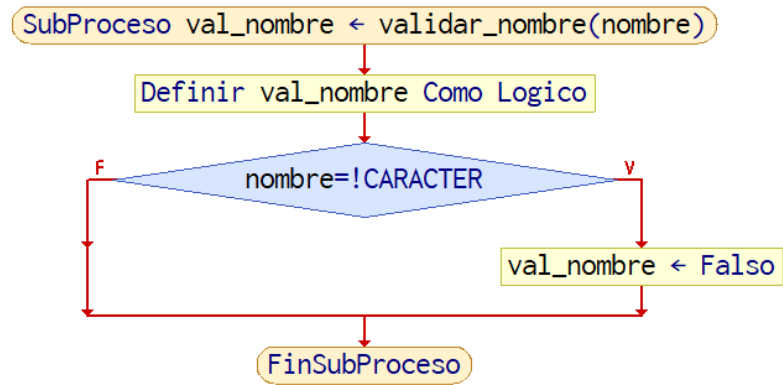
| | |
|---------|---|
| Name | RF4: Validar sobrenombre |
| Summary | La aplicacion valida que el sobrenombre no inicie con numeros |
| Inputs | |
| | sobrenombre |

| | |
|---------|--|
| Name | RF5: Validar clave |
| Summary | La aplicacion valida que la clave contenga AL MENOS uno de los simbolos obligatorios |
| Inputs | |
| | clave |

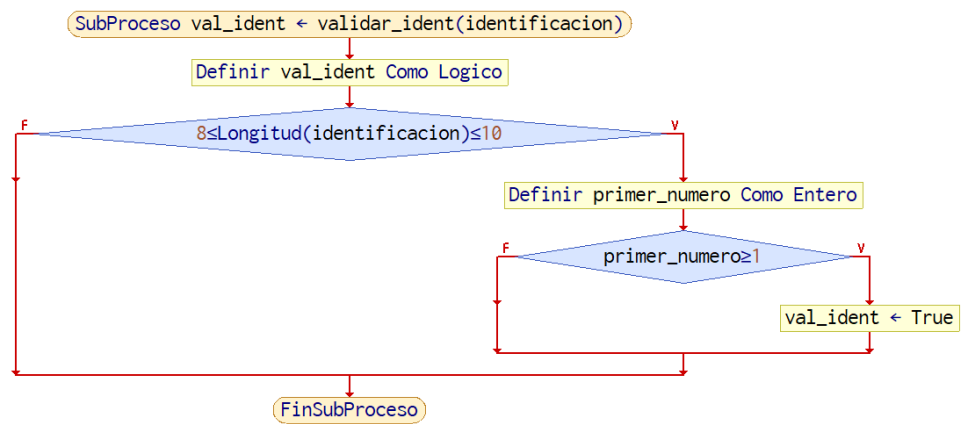
| | |
|---------|---|
| Name | RF6: Asignar Zona |
| Summary | La aplicacion asigna una zona segun el tiempo, tratamiento y conocimiento |
| Inputs | |
| | tiempo(int), tratamiento(str), conocimiento(str) |

| | |
|---------|---|
| Name | RF7: Validar informacion |
| Summary | La aplicacion evalua si el registro es Exitoso o No Exitoso(Si alguna de las validaciones da falsa) |
| Inputs | Recibe los booleanos de las funciones encargadas de validar |
| | val_nombre(bool) val_ident(bool) val_correo(bool) val_sobrenombre(bool) val_clave(bool) |

- Algoritmo para cada requisito



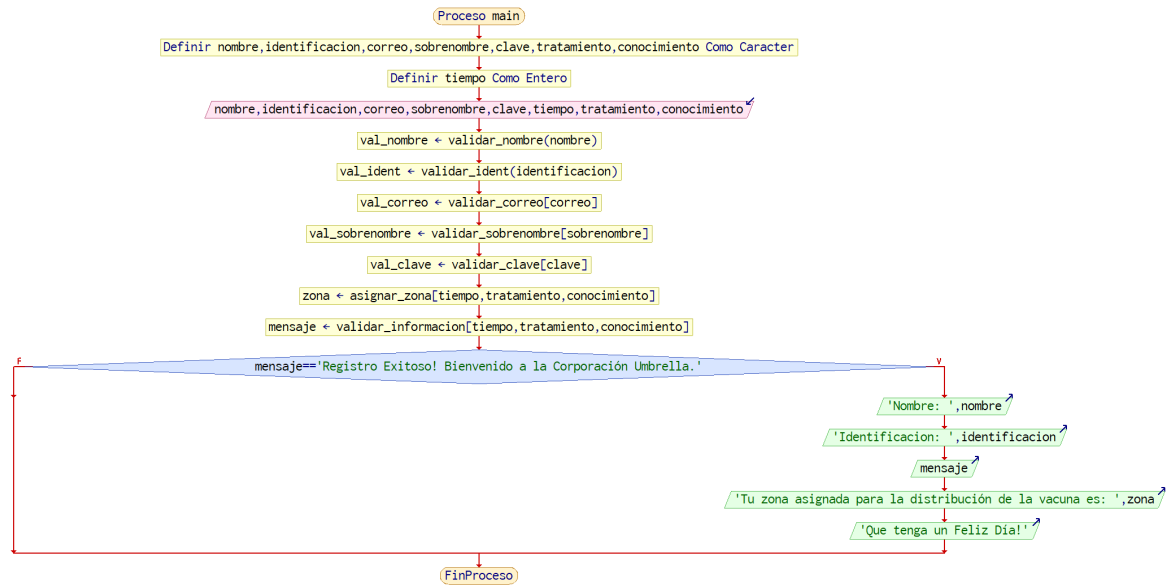
algoritmo validar_nombre(nombre)



algoritmo validar_ident(identificacion)

pd: los otros estan largos y la logica es similar, pero por cuestiones de tiempo dejemos hasta aqui 😊

- Algoritmo General



Codigo de la solucion → [Link al replit](#)

```

import registro as reg

# =====
# ESPACIO DE TRABAJO ALUMNO
# =====

def solicitar_datos():
    """solicitar la información al usuario, hacer uso de las funciones alojadas en el script
    registro.py para
    validar la información, y finalmente mostrar al usuario el mensaje especificado al final del
    archivo README.md
    """
    nombre = input().strip()
    identificacion = input()
    correo = input()
    sobrenombre = input()
    clave = input()
    tiempo = int(input())
    tratamiento = input().upper()
    conocimiento = input().upper()

    val_nombre = reg.validar_nombre(nombre)
    val_ident = reg.validar_ident(identificacion)
    val_correo = reg.validar_correo(correo)
    val_sobrenombre = reg.validar_sobrenombre(sobrenombre)
    val_clave = reg.validar_clave(clave)
    zona = reg.asignar_zona(tiempo, tratamiento, conocimiento)
    mensaje = reg.validar_informacion(
        val_nombre,
        val_ident,
        val_correo,
        val_sobrenombre,
        val_clave)

    # Si el registro es exitoso entonces... imprime
    if mensaje == "Registro Exitoso! Bienvenido a la Corporación Umbrella.":
        print(f"Nombre: {nombre}")
        print(f"Identificación: {identificacion}")
        print(mensaje)
        print(f"Tu zona asignada para la distribución de la vacuna es: {zona}")
        print("Que tenga un Feliz Día!")
    else:
        print(mensaje) # Mensaje de error

solicitar_datos()

```

main.py

[illegible]

registro.py

▼ Código registro.py

```
def validar_nombre(nombre):
    """ Valida si el nombre es string o no """

    Args:
        nombre(str)

    Return:
```

```

    bool

Examples
-----
>>> validar_nombre('Anita')
True

>>> validar_nombre('Pedro40')
False
"""
val_nombre = True
if not isinstance(nombre, str):
    val_nombre = False
if nombre.isalpha() == False:
    val_nombre = False
return val_nombre

def validar_ident(identificacion):
    """ Valida si la identificacion cumple con:
    Longitud(Valores entre 10'000.000 y 100'000.000)

    Args:
        identificacion(str)

    Return:
        bool

    Examples
    -----
    >>> validar_ident('01234567')
    False

    >>> validar_ident('1234567')
    False

    >>> validar_ident('1234567910')
    True
    """
    val_ident = False
    if 8 <= len(identificacion) <= 10:
        primer_numero = int(identificacion[0])
        if primer_numero >= 1:
            val_ident = True
    else:
        val_ident = False
    return val_ident

def validar_correo(correo):
    """ Valida si el correo contiene el caracter '@'
    y que no se repita mas de una vez

    Args:
        correo(str)

    Return:
        bool

    Examples
    -----
    >>> validar_correo('@')
    True

    >>> validar_correo('@@')
    False

    >>> validar_correo('santiagomatgmaildotcom')
    False
    """
    val_correo = False

```



```

if correo.rfind("@") < 0:
    val_correo = False
elif correo.count("@") >= 2:
    val_correo = False
else:
    val_correo = True
return val_correo

def validar_sobrenombre(sobrenombre):
    """ Valida que el sobrenombre no inicie con
    numeros

    Args:
        sobrenombre(str)

    Return:
        bool

    Examples
    -----
    >>> validar_sobrenombre('sanurb')
    True

    >>> validar_sobrenombre('6santi')
    False
    """
    val_sobrenombre = False
    if sobrenombre[0].isdigit():
        val_sobrenombre = False
    elif sobrenombre.count("a") > 3:
        val_sobrenombre = False
    else:
        val_sobrenombre = True
    return val_sobrenombre

def validar_clave(clave):
    """ Valida que la clave contenga AL MENOS uno de
    los simbolos obligatorios

    Args:
        clave(str)

    Return:
        bool

    Examples
    -----
    >>> validar_clave('clave123_')
    True

    >>> validar_clave('clave123')
    False
    """
    val_clave = False
    simbolos_obligatorios = ["_", "?", "&"]
    # Usamos una list comprehension para comprobar
    # entiendase ele como elemento
    comprobacion = [ele for ele in simbolos_obligatorios if (ele in clave)]
    # comprobamos si la cadena clave contiene algun simbolos obligatorio
    if bool(comprobacion) == False:
        val_clave = False
    else:
        val_clave = True
    return val_clave

def asignar_zona(tiempo, tratamiento, conocimiento):
    """ Asigna una zona segun el tiempo, tratamiento y conocimiento

```

```

Args:
    tiempo(int)
    tratamiento(str)
    conocimiento(str)
Return:
    str

Examples
-----
>>> asignar_zona(72, no, NO)
"Sur Occidente"

>>> asignar_zona(60, No, si)
"Norte"

>>> asignar_zona(80, si, no)
"Central"

"""
zona = ""
if 12 > tiempo >= 60:
    if tratamiento == "NO" and conocimiento == "SI":
        zona = "Norte"
    elif tratamiento == "SI" and conocimiento == "NO":
        zona = "Sur"
elif tiempo > 60:
    if tratamiento == "NO" and conocimiento == "SI":
        zona = "Oriente"
    elif tratamiento == "SI" and conocimiento == "SI":
        zona = "Occidente"
    elif tratamiento == "NO" and conocimiento == "NO":
        zona = "Sur Occidente"
else:
    zona = "Central"
return zona

def validar_informacion(
    val_nombre,
    val_ident,
    val_correo,
    val_sobrenombre,
    val_clave):
    """ Evalua si el registro es Exitoso o No Exitoso

Args:
    val_nombre(bool)
    val_ident(bool)
    val_correo(bool)
    val_sobrenombre(bool)
    val_clave(bool)
Return:
    str

Examples
-----
>>> validar_informacion(True, True, True, True, True)
"Registro Exitoso! Bienvenido a la Corporación Umbrella."

>>> validar_informacion(True, False, True, True, True)
"Registro No Exitoso, ident incorrecto."
"""
result = val_nombre and val_ident and val_correo and val_sobrenombre and val_clave
mensaje = ""
parametro = "" # Indica la primer ocurrencia de error

if not val_clave:
    parametro = "clave incorrecto"

```

```

elif val_sobrenombre == False:
    parametro = "sobrenombre incorrecto"

elif val_correo == False:
    parametro = "correo incorrecto"

elif val_ident == False:
    parametro = "ident incorrecto"

elif val_nombre == False:
    parametro = "nombre incorrecto"
# Validacion de criterios
if not result:
    mensaje = f"Registro No Exitoso, {parametro}."
else:
    mensaje = "Registro Exitoso! Bienvenido a la Corporación Umbrella."
return mensaje

```

▼ Código main.py

```

import registro as reg

# =====
# E S P A C I O   D E   T R A B A J O   A L U M N O
# =====

def solicitar_datos():
    """solicitar la información al usuario, hacer uso de las funciones alojadas en el script registro.py para
    validar la información, y finalmente mostrar al usuario el mensaje especificado al final del archivo  README.md
    """
    nombre = input().strip()
    identificacion = input()
    correo = input()
    sobrenombre = input()
    clave = input()
    tiempo = int(input())
    tratamiento = input().upper()
    conocimiento = input().upper()

    val_nombre = reg.validar_nombre(nombre)
    val_ident = reg.validar_ident(identificacion)
    val_correo = reg.validar_correo(correo)
    val_sobrenombre = reg.validar_sobrenombre(sobrenombre)
    val_clave = reg.validar_clave(clave)
    zona = reg.asignar_zona(tiempo, tratamiento, conocimiento)
    mensaje = reg.validar_informacion(
        val_nombre,
        val_ident,
        val_correo,
        val_sobrenombre,
        val_clave)

    # Si el registro es exitoso entonces... imprime
    if mensaje == "Registro Exitoso! Bienvenido a la Corporación Umbrella.":
        print(f"Nombre: {nombre}")
        print(f"Identificación: {identificacion}")
        print(mensaje)
        print(f"Tu zona asignada para la distribución de la vacuna es: {zona}")
        print("Que tenga un Feliz Día!")
    else:
        print(mensaje) # Mensaje de error

solicitar_datos()

```

Pruebas

- El reto se considera aprobado pasando exitosamente mínimo 4 de las 7 pruebas automáticas (tests)

Puedo ejecutar las pruebas apartir del docstring con el siguiente comando en el shell. En mi caso el archivo tiene por nombre `registro.py`.

```
python -m doctest registro.py -v
```

```
Console Shell Markdown
File "/home/runner/Reto-Semana-3-S4ntiago/registro.py", line 198, in registro.validar_informacion
Failed example:
  validar_informacion(True, True, True, True, True)
Expected:
  "Registro Exitoso! Bienvenido a la Corporación Umbrella."
Got:
  'Registro Exitoso! Bienvenido a la Corporación Umbrella.'
Trying:
  validar_informacion(True, False, True, True, True)
Expecting:
  "Registro No Exitoso, ident incorrecto."
*****
File "/home/runner/Reto-Semana-3-S4ntiago/registro.py", line 201, in registro.validar_informacion
Failed example:
  validar_informacion(True, False, True, True, True)
Expected:
  "Registro No Exitoso, ident incorrecto."
Got:
  'Registro No Exitoso, ident incorrecto.'
Trying:
  validar_nombre('Anita')
Expecting:
  True
ok
Trying:
  validar_nombre('Pedro40')
Expecting:
  False
ok
Trying:
  validar_sobrenombre('sanurb')
Expecting:
  True
ok
Trying:
  validar_sobrenombre('6santi')
Expecting:
  False
ok
1 items had no tests:
  registro
5 items passed all tests:
  2 tests in registro.validar_clave
  3 tests in registro.validar_correo
  3 tests in registro.validar_ident
  2 tests in registro.validar_nombre
  2 tests in registro.validar_sobrenombre
*****
2 items had failures:
  3 of 3 in registro.asignar_zona
  2 of 2 in registro.validar_informacion
17 tests in 8 items.
12 passed and 5 failed.
***Test Failed*** 5 failures.
~/Reto-Semana-3-S4ntiago$
```

Resultado de las pruebas usando módulo *doctest* de Python.

Input/Output Tests

▶ Run tests

*2registro:

Passed

Results

*4registro:

Passed

Results

*5registro:

Passed

Results

*7registro:

Failed

Results

*1registro:

Passed

Results

*6registro:

Passed

Results

*3registro:

Passed

Results

Ejecutando las pruebas que vienen por defecto

Estuvo chevere el ejercicio, sin embargo el hubo error por parte de quien programo la prueba #7, ya lo verifique con un tutor. talvez se vea mejor en esta pagina web que en este pdf. aqui el [link de notion](#)

